Appendix F - MAPI Technical Report

What Does MAPI Do?

Eudora's MAPI support allows users to quickly attach documents to e-mail messages directly from the application that created the document. Without MAPI, users must first save the document, remember what folder the document is in, switch to Eudora, and then remember to manually attach the document to the outgoing message.

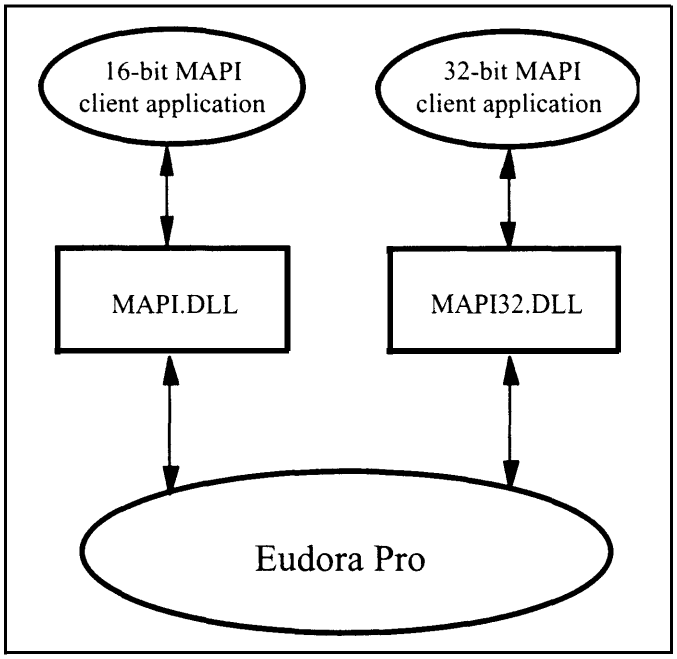
MAPI streamlines this process dramatically. To e-mail the current, open document from your word processor, select the Send command from your word processor's File menu. This automatically activates Eudora and attaches a snapshot of the open document to a new composition message.

The MAPI system standardizes how messages are handled by client applications so that each client application does not have to have custom code for each target messaging application. MAPI accomplishes this by providing a standard *application program interface* used by all MAPI-enabled client applications.

An additional MAPI feature supported by Microsoft Office applications is the ability to add a "routing slip" to a Word, Excel, or PowerPoint document. This routing slip contains a list of e-mail recipients obtained from the MAPI subsystem. Once a document has an embedded routing slip, then it can be semi-automatically routed as an attachment via e-mail to all recipients listed in the routing slip. Once the routing is complete, the annotated document is returned back to the original sender.

MAPI Overview

Let's start with a picture:


16-bit MAPI client application
32-bit MAPI client application
MAPI.DLL
MAPI32.DLL
Eudora Pro

A MAPI *client application* is any 16-bit or 32-bit Windows application that knows how to access the standard MAPI messaging functions in a library known as a DLL (Dynamic Link Library). The functions in the MAPI DLL allow a MAPI client application to transparently and generically access a MAPI *service provider.* A MAPI service provider is the application that handles the receipt, transmission, and storage of messages. Examples of MAPI client applications ("front-ends") include Microsoft Word and Microsoft Excel. Examples of MAPI service providers ("back-ends") include Microsoft Mail, Microsoft Exchange, and Microsoft Fax.

All 16-bit client applications use the 16-bit MAPI.DLL and all 32-bit client applications use the 32-bit MAPI32.DLL. The MAPI and MAP132 DLLs are "twins" which contain the same list of MAPI functions—they are parallel implementations of the 16-bit and 32-bit MAPI functions. These DLLs are provided by Microsoft as standard components of Windows for Workgroups, Windows 95, and Windows NT. For non-networked Windows 3.x, the MAPI.DLL is available in add-on development kits or is included with MAPI-enabled applications. The MAPI DLLs are normally installed in the Windows SYSTEM directory (or SYSTEM32 for Windows NT).

As shown in the diagram on the previous page, when a MAPI client application wishes to send a document, it simply loads the appropriate MAPI library (DLL) and calls the defined MAPI functions. The MAPI DLL takes care of routing the messaging and authentication requests to the appropriate MAPI service provider application, displaying the address book user interface, and returning address book and messaging data to the MAPI client application. The MAPI DLL also provides an optional user interface for user authentication. For example, the user may need to supply a user name and password to the mail system in order to "log on" to the mail system. (The Eudora implementation of MAPI does not implement authentication since Eudora itself requires authentication to access the POP3 server.)

Eudora Pro Implementation of MAPI

Eudora Pro implements a subset of the full MAPI library by providing two "replacement DLLs" for the standard Microsoft MAPI DLLs. The Eudora EUMAPI.DLL is a replacement for the 16-bit Microsoft MAPI.DLL and the Eudora EUMAPI32.DLL is a replacement for the 32-bit Microsoft MAPI32.DLL. The Eudora MAPI DLLs must be located in the same directory as the Eudora program.

The Eudora MAPI DLLs implement the standard *Simple MAPI* functions detailed in the MAPI specification. The MAPI specification also defines *Extended MAPI* functions, however, the Eudora MAPI DLLs implement only the Simple MAPI subset.

Note: The Eudora MAPI implementation requires all MAPI client applications to use only the Simple MAPI functions supported by the Eudora MAPI DLLs.

MAPI client applications which use only the basic Simple MAPI calls will generally not be able to tell the difference between the Eudora MAPI DLL functions and the Microsoft MAPI DLL functions.

It is important to understand that MAPI client applications load the MAPI DLL libraries at runtime whenever they need to access the MAPI functions. Each client application expects to find either the 16-bit MAPI.DLL file or the 32-bit MAPI32.DLL file in a common, application-independent location (generally the Windows SYSTEM directory). Therefore, it is not sufficient to copy the EUMAPI.DLL and EUMAPI32.DLL Eudora DLL files into the Windows SYSTEM directory alongside the standard Microsoft MAPI.DLL and MAPI32.DLL files. For client applications to find the Eudora MAPI DLLs, the DLL files must be named MAPI.DLL and MAPI32.DLL. This creates a conflict since most Windows installations will have the MAPI.DLL and MAPI32.DLL files preinstalled in the Windows SYSTEM directory to support Microsoft Mail (Windows for Workgroups) or Microsoft Exchange (Windows 95, Windows NT). Therefore,

Note: Eudora Pro is able to swap the Eudora EUMAPI and EUMAPI32 DLLs with the Microsoft MAPI and MAPI32 DLLs when the user launches Eudora Pro, and is able to unswap the Eudora MAPI DLLs when the user exits Eudora Pro.

This approach gives the user the most flexibility and preserves the user's ability to use Microsoft Mail and/or Microsoft Exchange when Eudora is not running. If we "permanently" install the Eudora MAPI DLLs over the existing Microsoft MAPI DLLs, then applications (such as the Microsoft Fax service bundled with Microsoft Exchange) which rely on the Microsoft MAPI DLLs will no longer work. This is clearly unacceptable for users who need to use MAPI for both Microsoft Exchange and Eudora.

Eudora MAPI Startup Procedure

When launched, Eudora Pro runs the following “swap” procedure when the user has selected either the “Always” or the “When Eudora is running” MAPI Server option in Eudora (see Tools/ Options/ MAPI):

1. Check to see whether or not the Eudora MAPI DLLs are already installed in the Windows SYSTEM directory. If so, then finish.
2. Check for existing Microsoft MAPI.DLL and MAP132.DLL files. If found, rename MAPI.DLL to MAPI.000 and rename MAPI32.DLL to MAP132.000. (If a MAPI.000 file already exists, then Eudora uses MAPI.001, MAPI.002 etc.)
3. Copy the EUMAPI.DLL and EUMAPI32.DLL files from the Eudora program directory to the Windows SYSTEM directory as MAPI.DLL and “ MAPI32.DLL, respectively.

Eudora MAPI Shutdown Procedure

When shutdown, Eudora Pro runs the following “unswap” procedure when the user selects either the “When Eudora is running” or “Never” MAPI Server option in Eudora (see Tools/ Options/ MAPI):

1. Check to see whether or not the Eudora MAPI DLLs are already installed in the Windows SYSTEM directory. If not, then finish.
2. Delete the Eudora MAPI.DLL and MAPI32.DLL files.
3. Rename the MAPI.000 and MAP132.000 files, if any, to MAPI.DLL and MAPI32.DLL, respectively. (If a MAPI.001, MAPI.002 etc file exists, then Eudora renames the one with the highest number.)

Eudora DLL Swapping Restrictions

It is important to note that there are several restrictions with the above Eudora swap and unswap procedures: The Eudora Pro swap and unswap procedures can only run successfully if the MAPI.DLL and MAPI32.DLL are not currently "in use" by one or more MAPI client applications.

When a MAPI client application loads a MAPI or MAP132 DLL file, Windows "locks" the DLL file while the library is loaded into memory to show that the file is "in use." Eudora can normally detect that the MAPI.DLL and/or MAPI32.DLL files are "in use." If Eudora detects that a MAPI or MAP132 DLL is locked, it displays an error message and skips the swap or unswap procedure.

When Eudora is forced to skip the swap or unswap procedure, this means that the MAPI DLLs are in the wrong "state" with respect to Eudora -- that is, 1) the Microsoft MAPI DLLs could be installed even after Eudora starts, or 2) the Eudora MAPI DLLs could be installed even after Eudora shuts down. To prevent this from happening, use the following procedure when using Eudora MAPI:

1. Start Windows.
2. Start Eudora Pro.
3. Start any MAPI client applications.
4. Send attachments to Eudora via the installed Eudora MAPI interface.
5. Shutdown all MAPI client applications.
6. Shutdown Eudora Pro.
7. Exit Windows.

Once Eudora's MAPI DLLs get into the wrong "state" with respect to Eudora, you cannot correct the state mismatch until all MAPI client applications unload the MAPI DLLs and Windows is able to unlock the DLL file. Since the MAPI DLLs are shared by multiple MAPI client applications, Windows does not unlock the MAPI DLL file until the last MAPI client application is shut down. Therefore, to force all MAPI client applications to unload the DLLs, you must shutdown all MAPI client applications.

Important: When running 16-bit MAPI client applications under Windows 3.1 or Windows NT, then Eudora cannot detect the lock placed on the MAPI DLLs by Windows unless the SHARE program is running. This means that Eudora can inadvertently perform the DLL swap and/or unswap procedures while the MAPI DLL is loaded into memory. This almost always causes Windows to become unstable and can lead to crashes in MAPI client applications as well as in Windows itself.

Note: If you run 16-bit MAPI client applications under Windows 3.1 or Windows NT, then you should always run the SHARE program.

The good news is that Windows for Workgroups and Windows 95 implement the SHARE functionality without requiring you to explicitly run the SHARE program. By default, Windows NT only implements the SHARE functionality for 32-bit applications. If you are running a 16-bit application under Windows NT, then you must run the SHARE program explicitly as you do under Windows 3.1.